

# Modeling Information Systems in UML

**Peter Rittgen**

*University College of Borås, Sweden*

## INTRODUCTION

The first approaches to object-oriented modeling appeared by the second half of the 1970s, but not much happened for more than a decade, so there were still barely more than a handful of modeling languages at the end of the 1980s. It was the early 1990s that witnessed an ever-growing market in competing object-oriented methods so that potential users found it increasingly difficult to identify any single method that suited their needs. This phenomenon came to be known as the “method wars.” Towards the end of 1994 two of the “big” players, Grady Booch and Jim Rumbaugh, decided to join forces by integrating their respective approaches, the Booch method and OMT (Object Modeling Technique). In late 1995, Ivar Jacobson became a member of this team merging in his OOSE method (Object-Oriented Software Engineering). The efforts of the “three amigos” aimed at overcoming unnecessary differences between the individual approaches and also improving each of them by creating a common, standardized modeling language that could serve as an industry standard. The result was the release of the Unified Modeling Language (UML), version 0.9, in June 1996. The UML partners, an industry consortium, performed further work on UML. This led to the versions 1.0 and 1.1 being introduced in 1997. The latter was adopted by the OMG (Object Management Group) in the same year. The current version is 1.5 (OMG, 2003) but a major upgrade to 2.0 is in preparation (Björkander & Kobryn, 2003).

## BACKGROUND

UML is a language to support the development of software systems. It features a common framework, which provides a set of diagram types covering different aspects of an information system. Here we will focus on the ones we deem to be most important: class diagram, use case diagram, and activity diagram. The first is the principal diagram for the static view on a system. The second is often put forward as the central diagram in communication with the user (see, e.g., Dobing & Parsons, 2000). The latter plays a central role in the information system (or business-oriented) perspective (see following sections).

Elementary concepts of the UML are actor, activity (and state), object, and class.

An **actor** is a human being or a (computer) system who/which is able to perform activities on his/her/its own. The actor typically represents a group of similar human beings/systems and corresponds to the role the actor performs in the given context. *Teacher* is an example for such an actor. In UML actors are shown as stick figures.

An **activity** refers to a logically connected set of actions that are carried out as a unit in some order. These actions might be executed sequentially, alternatively, concurrently or in any combination thereof. *Grade exam* is an example for an activity.

“An **object** is an abstraction of a set of real-world things such that:

- all of the real-world things in the set - the instances - have the same characteristics,
- all instances are subject to and conform to the same rules” (Shlaer & Mellor, 1988, p. 14).

The structure (attributes) and behavior (operations) of similar objects are gathered in their common **class**. The values of the attributes at a certain point in time represent the state of the object. Classes are drawn as rectangles with the object identifier printed in bold face. Additional horizontal compartments can be introduced for the attributes and operations of the class. The object is depicted in the same way with the object identifier underlined (optionally followed by a colon and the respective class identifier). *Grade* is an example of a class, *F: Grade* that of an object.

*Figure 1* shows how these concepts are related to each other: actors perform activities that involve objects. Each object is an instance of precisely one class. The class diagram shows classes and their relations (called associations). An activity diagram gives a detailed account of the order in which activities are executed. It can also show the relevant states of the involved objects. The use case diagram visualizes use cases (complex activities) and their relation to actors.

## Use Case Diagram

Use cases were introduced in 1992 by Jacobson and his colleagues. Their book is now available in the 2<sup>nd</sup> edition

Figure 1. Language overview

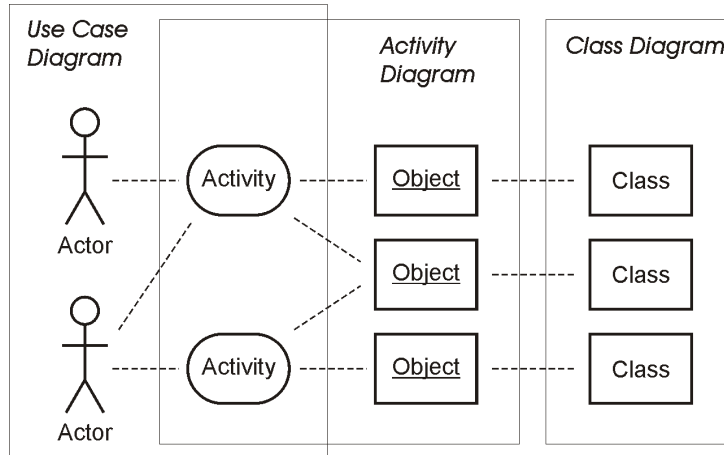
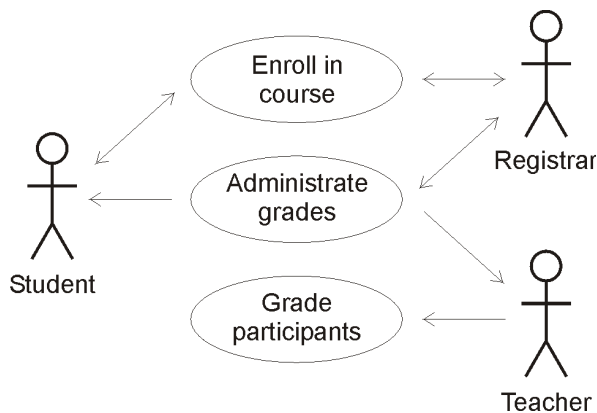


Figure 2. A use case diagram



(Christerson, Jonsson, & Övergaard, 2004). A use case describes a way in which an actor can interact with your organization. It forms part of a use case scenario, a business situation that is supposed to be supported by the information system in question. Figure 2 gives an example of a use case diagram involving 3 use cases and 3 actors.

The arrows indicate the direction of the information flow. So the registrar can both read and enter/change grades, whereas teachers and students can only read them. Often the arrowheads are omitted to simplify modeling. Each use case is detailed by a description in structured English of the activities that make up this use case. The use case diagram and the use cases form the basis for the development of class diagrams and activity diagrams. The former specify the static structure of the information that is handled by the use cases, and the latter give a precise, formalized account of the process logic.

### Activity Diagram

An activity diagram consists of the detailed activities that make up the use cases. Figure 3 gives an example of such a diagram in relation to the use cases of Figure 2.

The use case *Enroll in course* comprises the activities *Apply for course* (performed by the student) and *Enroll student* (performed by the registrar). *Grade participants* maps to *Grade exam* assuming that the grade for the course is determined by a final exam only. The use case *Administrate grades* involves the activities *Enter grade*, *Check grade* and *Change grade*. Note that the activity diagram also contains activities that are not present in the use case diagram, such as *Deliver course* and *Write exam*. That is because, in our example, they are not supposed to be supported by the information system and hence do not constitute use cases of such a system. But they are still an important part of the overall business process.

The business process in Figure 3 starts with the initial state (the black dot). The activities are in the boxes with round sides. The rectangles contain objects-in-state where the object identifier is underlined and the object's state is enclosed in square brackets. So after *Enter grade*, for example, the object *Grade* is in state *entered*. Alternative paths leaving an activity must be labeled by so-called guards that determine under which condition each path is taken. These guards are also enclosed in square brackets and should be mutually exclusive. The process terminates when the final state is reached (i.e., the circle containing the black dot). Actors are included in the form of so-called swim lanes. Concurrent execution is achieved by introducing synchronization bars. For more detailed information on activity diagrams we refer the reader to the OMG (2003, p. 3-155 ff).

Figure 3. An activity diagram

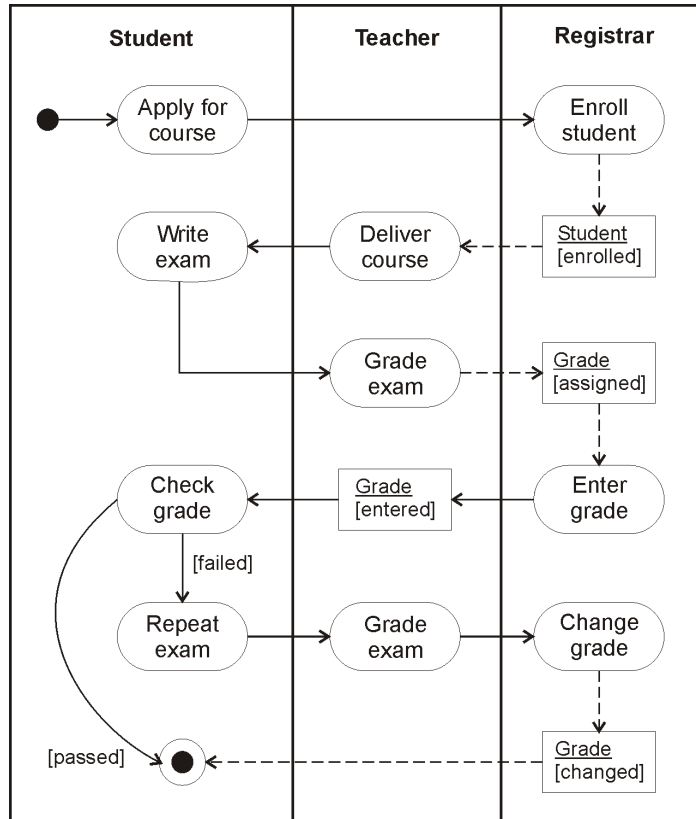
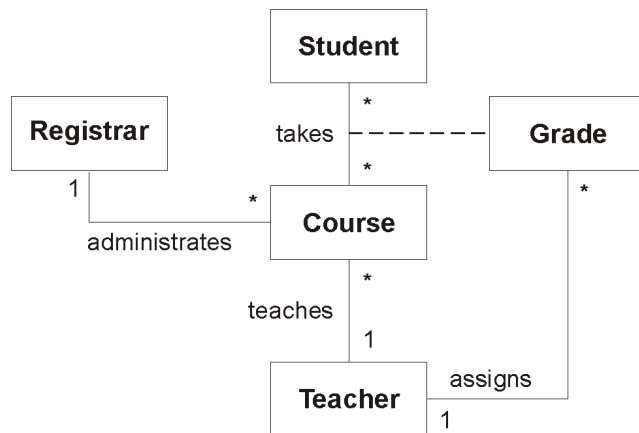


Figure 4. A class diagram



### Class Diagram

A typical approach to identifying classes is that of looking at the nouns contained in a textual specification of the system. Jacobson, Christerson, Jonsson, & Övergaard (2004) suggest that we should limit our attention to the

nouns contained in the Use Case Diagram instead to avoid having to eliminate inappropriate class candidates. If we apply this to the use case diagram of Figure 2 we arrive at the class diagram of Figure 4 where all the nouns of the former map to a class of the latter. Observe that the noun *participants* maps to the class *Student*.

The class diagram also shows associations between the classes that indicate the way in which they are related to each other. A single line is drawn connecting the respective classes. It can be labeled with a verb that determines the nature of the association. In addition we can also specify the cardinality with the asterisk referring to an unlimited number. In the example a *Teacher teaches* a number of *Courses* but each *Course* is only taught by one *Teacher*. *Grade* is a so-called association class. It qualifies the relation between *Student* and *Course* and is therefore attached to the association path by a dashed line. For further details refer to the OMG (2003, p. 3-34 ff).

## MODELING INFORMATION SYSTEMS IN UML

UML has been designed primarily to support the development of software systems. In such systems the soft-

ware artifact plays the central role and the role of human beings is restricted to the specification of requirements and, later on, the use of the system. An information system, on the other hand, is a human activity system (sometimes also called a socio-technical system) where some part of the system *might* be supported by software (Buckingham, Hirschheim, Land, & Tully, 1987).

The focus being on software systems, it is natural that UML does not emphasize the aspects of an information system that aims at the value and support it can provide to the business, such as strategy (e.g., value chains & strategic goals) and organization (e.g., organizational charts and business processes). These issues are dealt with in “business modeling” (also called enterprise modeling) but UML is beginning to advance into that field, too. The OMG (2003, pp. 1-9), for instance, claims that business processes can be represented in UML. This claim can, however, be challenged in a number of ways. First of all there is very little direct support for business processes in the basic UML diagrams (Bruno, Torchiano, & Agarwal, 2002). Instead many of the required concepts are “hidden” in extensions to UML, such as Enterprise Collaboration Architecture (ECA) (OMG, 2004). Another issue is whether UML models are indeed “understandable” enough to form a solid basis for the communication between “developers” and “users.” An empirical study of UML’s usability (Agarwal & Sinha, 2003) showed that UML diagrams score between 4.3 and 5.3 on a 7-point Likert scale, that is, the ease-of-use of UML is closer to indifference (4) than to the optimum (7). Understandability of UML is also weakened by construct redundancies, inconsistencies and ambiguities (Shen & Siau, 2003).

Candidate languages for business processes (in native UML) are use cases and activity diagrams. The former have been criticized for their conceptual flaws (Dobing & Parsons, 2000), such as fragmentation of objects, and for construct overload and ambiguity (Shen & Siau, 2003). In addition they restrict the specification of process logic to a textual form, which is neither rigorous nor expressive enough for typical flows of control. Activity diagrams come closer to fulfilling the requirements of a business-process language and are therefore often used in that way (see next section).

## **FUTURE TRENDS**

So far, there is still an abundance of languages used for business modeling, for example, Architecture of Integrated Information Systems ARIS (Scheer, 1999), Open System Architecture for Computer-Integrated Manufacturing CIMOSA (AMICE, 1993), Integrated Definition IDEF\* (Ang, Pheng, & Leng, 1999), Integrated Enterprise Modeling IEM (Spur, Mertins, & Jochem, 1995), Dynamic

Essential Modeling of Organization DEMO (Reijswoud, Mulder, & Dietz, 1999), and GRAI Integrated Methodology GIM (Doumeingts, 1998). Often modelers also pick out particular model types from integrated methods or use “stand-alone” languages that were developed for specific purposes. In the area of business process modeling these include: Event-Driven Process Chains (EPC) (from ARIS), Business Process Modeling Language (Arkin, 2002), IDEF3 (Mayer et al., 1997), Tropos (Castro, Kolp, & Mylopoulos, 2002), and Petri Nets (Aalst, Desel, & Oberweis, 2000).

But how can we bridge the gap between these enterprise-modeling languages and UML? One approach consists of devising mechanisms that “translate” domain models into UML models. Such a translation is by no means straightforward but typically involves the reconstruction of the domain model as a UML diagram by a modeler who is experienced in both the domain language and UML. It usually involves the loss of information that was present in the domain model and the necessity to add information not yet represented. This implies also the risk of introducing errors, or more precisely inconsistencies between the original and the translated model. Examples of such an approach can be found in many of the languages mentioned due to the popularity of UML, for example, EPC to UML (Nüttgens, Feld & Zimmermann, 1998), IDEF and UML (Noran, 2000), Tropos and UML (Mylopoulos, Kolp, & Castro, 2001), DEMO and UML (Mallens, Dietz, & Hommes, 2001), or Petri Nets and UML (Gou, Huang, & Ren, 2000).

Another – and perhaps even more promising – approach is to equip UML itself with domain support. This can be done in at least two ways: extending existing concepts or introducing new ones (typically in the form of profiles). A major effort towards the latter is being made under the heading “Enterprise Distributed Object Computing (EDOC)” and was adopted by the OMG in February 2004 (version 1.0) but work is still ongoing. Its core is the Enterprise Collaboration Architecture ECA (OMG, 2004). The former approach is followed by several researchers independently. In the area of business processes and workflows they primarily investigate extensions of use cases and activity diagrams (see, for example, Rittgen, 2003; Dumas & Hofstede, 2001).

## **CONCLUSION**

The UML is a widely used modeling language in the area of software systems modeling but it still has to gain ground in domain modeling, especially in business (or enterprise) modeling. Promising steps have already been taken that have so far led to results such as EDOC. But there is still a need for research that provides the core

concepts and diagrams of the UML with domain-specific semantics to make them more useful for business modeling. On the other hand, add-on concepts such as EDOC processes require a tighter semantical integration with the existing models. Future research and development will show which of the paths towards integration is more promising and will hence be taken. But all approaches mentioned share a common goal: to improve the communication and understanding between the different people involved in shaping an information system and thereby also improving the system itself.

## REFERENCES

- AMICE. (1993). *CIMOSA: Open system architecture for CIM* (2<sup>nd</sup> rev. and ext. ed.). Berlin: Springer.
- Agarwal, R., & Sinha, A.P. (2003). Object-oriented modeling with UML: A study of developers' perceptions. *Communications of the ACM*, 46 (9), 248-256.
- Ang, C.L., Pheng, K.L., & Leng, G.R.K. (1999). IDEF\*: A comprehensive modelling methodology for the development of manufacturing enterprise systems. *International Journal of Production Research*, 37 (17), 3839-3858.
- Arkin, A. (2002). *Business process modeling language*. Aurora, CO: BPMI.org.
- Björkander, M., & Kobryn, C. (2003). Architecting systems with UML 2.0. *IEEE Software*, 20 (4), 57-61.
- Bruno, G., Torchiano, M., & Agarwal, R. (2002). UML enterprise instance models. In S. Iyer & S. Naik (eds.), *CIT 2002: Proceedings of the Fifth International Conference on Information Technology*. Bhubaneswar, India, December 21-24, 2002. New Delhi, India: Tata McGraw-Hill.
- Buckingham, R.A., Hirschheim, R.A., Land, F.F., & Tully, C.J. (1987). Information systems curriculum: A basis for course design. In R.A. Buckingham, R.A. Hirschheim, F.F. Land, & C.J. Tully (eds.), *Information Systems Education. Recommendations and Implementation* (pp. 14-133). Cambridge: Cambridge University Press,.
- Castro, J., Kolp, M., & Mylopoulos, J. (2002). Towards requirements-driven information systems engineering: The Tropos project. *Information Systems*, 27 (6), 365-389.
- Dobing, B., & Parsons, J. (2000). Understanding the role of use cases in UML: A review and research agenda. *Journal of Database Management*, 11 (4), 28-36.
- Doumeings, G. (1998). GIM: GRAI integrated methodology. In A. Molina, A. Kusiaka, & J. Sanchez (eds.), *Handbook of Life Cycle Engineering: Concepts, Models and Methodologies* (pp. 227-288). Dordrecht, Netherlands: Kluwer Academic Publishers.
- Dumas, M., & Hofstede, A.H.M. (2001). UML activity diagrams as a workflow specification language. In M. Gogolla & C. Kobryn (eds.), *UML 2001: The Unified Modeling Language*. Lecture Notes in Computer Science. (Vol. 2185) (pp. 76-90). 4th International Conference, Toronto, Canada, October 1-5, 2001., Berlin: Springer.
- Gou, H., Huang, B., & Ren, S. (2000). A UML and Petri nets integrated modeling method for business processes in virtual enterprises. In S. Staab & D. O'Leary (eds.), *Bringing Knowledge to Business Processes*. Papers from 2000 AAAI Spring Symposium (pp. 142-144). Menlo Park, CA: AAAI Press.
- Jacobson, I., Christerson, M., Jonsson, P., & Övergaard, G. (2004). *Object-oriented software engineering: A use case driven approach* (2<sup>nd</sup> ed.). Wokingham, U.K.: Addison-Wesley.
- Mallens, P., Dietz, J., & Hommes, B.J. (2001). The value of business process modeling with DEMO prior to information systems modeling with UML. In J. Krogstie, K. Siau, & T. Halpin (eds.), *EMMSAD'01: 6<sup>th</sup> IFIP 8.1 Workshop on Evaluation of Modeling Methods in Systems Analysis and Design*. Oslo, Norway: SINTEF.
- Mayer, R.J., Menzel, C.P., Painter, M.K., deWitte, P.S., Blinn, T., Perakath, B., Sartor, J.M., & McManus, J.C. (1997). *Information integration for concurrent engineering (IICE): IDEF3 process description capture method report*. AL/HR-TP-1996-0030. College Station, TX: Knowledge Based Systems Inc. & Wright-Patterson AFB, OH: Armstrong Laboratory.
- McLeod, G. (2000). Beyond use cases. In K. Siau, Y. Wand, & A. Gemino (eds.), *EMMSAD'00: 5<sup>th</sup> IFIP 8.1 Workshop on Evaluation of Modeling Methods in Systems Analysis and Design*. Stockholm, Sweden, June 4-5, 2000.
- Mylopoulos, J., Kolp, M., & Castro, J. (2001). UML for agent-oriented software development: The Tropos proposal. In M. Gogolla & C. Kobryn (eds.), *UML 2001: The Unified Modeling Language*. Lecture Notes in Computer Science (vol. 2185) (pp. 422-441). 4<sup>th</sup> Int. Conference, Toronto, Canada, October 1-5, 2001., Berlin: Springer.
- Noran, O. (2000). *Business modelling: UML vs. IDEF*. Brisbane, Australia: Griffith University. Retrieved from [www.cit.gu.edu.au/~noran](http://www.cit.gu.edu.au/~noran).
- Nüttgens, M., Feld, T., & Zimmermann, V. (1998). Business process modeling with EPC and UML: Transformation or integration? In M. Schader & A. Korthaus (eds.), *The Unified Modeling Language: Technical Aspects and Applications* (pp. 250-261). Berlin: Springer.

OMG. (2003, March). *Unified Modeling Language Specification*. Version 1.5. Needham: OMG. Retrieved from [www.uml.org](http://www.uml.org).

OMG. (2004, February). *Enterprise collaboration architecture specification*. Version 1.0. Needham: OMG. Retrieved from [www.uml.org](http://www.uml.org).

Palanque, P., & Bastide, R. (2003). UML for interactive systems: What is missing. In G.W.M. Rauterberg, M. Menozzi, & J. Wesson(eds.), *Human-Computer Interaction INTERACT '03*. 9<sup>th</sup> IFIP TC13 International Conference on Human-Computer Interaction, September 1-5, 2003, Zürich, Switzerland. Amsterdam, Netherlands: IOS Press.

Rittgen, P. (2003). Business processes in UML. In L. Favre (ed.), *UML and the Unified Process* (pp. 315-331). Hershey, PA: IRM Press.

Scheer, A.W. (1999). *ARIS: Business process modeling*. Berlin: Springer.

Shen, Z., & Siau, K. (2003). An empirical evaluation of UML notational elements using a concept mapping approach. In S.T. March, A. Massey, & J.I. DeGross (eds.), *Proceedings of the 24<sup>th</sup> Int. Conference on Information Systems* (pp 194-206). Seattle, WA, December 14-17, 2003. Atlanta, GA: AIS.

Shlaer, S., & Mellor, S.J. (1988). *Object-oriented systems analysis: Modeling the world in data*. Englewood Cliffs, NJ: Prentice-Hall.

Spur, G., Mertins, K., & Jochem, R. (1995). *Integrated enterprise modeling*. Berlin: Beuth.

Van der Aalst, W., Desel, J., & Oberweis, A. (2000). *Business process management: Models, techniques and*

*empirical studies*. Lecture Notes in Computer Science (Vol. 1806). Berlin: Springer.

Van Reijswoud, V.E., Mulder, J.B.F., & Dietz, J.L.G. (1999). Speech act based business process and information modelling with DEMO. *Information Systems Journal*, 9 (2), 117-138.

## KEY TERMS

**Activity:** A logically connected set of actions that are carried out as a unit in some order. It is associated with a state (called action state) in which the system remains while the activity is performed.

**Actor:** A person or (computer) system that can perform an activity. The actor does not refer to a particular individual but rather to a role (e.g., *Teacher*).

**Attribute:** A property of an object/class. The class *Car*, for example, can have an attribute *Color*, its value for the object *MyCar*: *Car* might be *blue*.

**Class:** A template for similar objects defining attributes and operations.

**Object:** An abstraction of a set of real-world things that has state, behavior and identity. An instance of its class where the values of the attributes determine the state, and the operations the behavior.

**Operation:** A function or transformation that may be applied to or by objects in a class. The class *Account*, for example, might have the operations *Open*, *Close*, *PayIn* (*Amount*), and *Withdraw* (*Amount*).

**State:** A certain combination of attribute values of an object.